I hereby certify that this correspondence is being deposited with the U.S. Postal Service as Express Mail, Airbill No. EV 553 871 303 US, in an envelope addressed to: MS Appeal Brief - Patents, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on the date shown below.

Dated: November 15, 2005 Signature: (Kevin J. Canning)

Docket No.: MWS-072
(PATENT)

O I P E
NOV 1 5 2005
PATENT & TRADEMARK OFFICE

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of:
John Ciolfi

U.S. Serial No.: 09/911663

Confirmation No.: 3836

Filed: July 24, 2001

Group Art Unit: 2173

For: *Handling Parameters in Block Diagram Modeling*

Examiner: N. Pillai

APPEAL BRIEF

Mail Stop Appeal Brief - Patents
Commissioner for Patents
Post Office Box 1450
Alexandria, VA 22313-1450

Dear Sir:

In furtherance of the Notice of Appeal filed on August 15, 2005, this brief is filed with a one-month extension of time under 37 C.F.R. 1.136(a).

The fees required under § 41.20(b)(2) are dealt with in the accompanying TRANSMITTAL OF APPEAL BRIEF.

This brief contains items under the following headings as required by 37 C.F.R. § 41.37 and M.P.E.P. § 1206:

I.	Real Party In Interest
II	Related Appeals and Interferences
III.	Status of Claims
IV.	Status of Amendments

11/18/2005 DTESSEM1 00000021 120080 09911663

02 FC:1402 500.00 DA

I. REAL PARTY IN INTEREST

The real party in interest for this appeal is:

The MathWorks, Inc.

II. RELATED APPEALS, INTERFERENCES, AND JUDICIAL PROCEEDINGS

There are no other appeals, interferences, or judicial proceedings known to Appellant, the Appellant's legal representative, or assignee which will directly affect or be directly affected by or have a bearing on the Board's decision in this appeal.

III. STATUS OF CLAIMS

A. Total Number of Claims in Application

There are 31 claims pending in application.

B. Current Status of Claims

1. Claims canceled: 1-15
2. Claims withdrawn from consideration but not canceled: 0
3. Claims pending: 16-46
4. Claims allowed: 0
5. Claims rejected: 16-46

C. Claims On Appeal

The claims on appeal are claims 16-46

IV. STATUS OF AMENDMENTS

No amendment has been filed after the final rejection mailed on February 17, 2005 (Paper No. 9).

V. SUMMARY OF CLAIMED SUBJECT MATTER

The present invention is directed toward improving a graphical block diagram environment. Such an environment may be used for modeling systems and analyzing their behavior. System elements are presented as blocks, having inputs and/or outputs and, optionally, parameters that may influence the behavior of the block.

One embodiment of the invention provides a mapping between user-defined block parameters and a run-time version of the parameters to achieve optimal implementation of block equations in the execution (run-time) environment. In the present invention, the block equations use run-time parameter configurations derived from the parameters entered by the user for an optimal allocation of resources, such as memory space, in the execution of the block equations. The present invention may pool together the like non-interface run-time block parameters to reduce repetition of the non-interfaced run-time block parameters. A non-interfaced parameter is a parameter whose value does not change either during simulation or in generated code. The pooling aspect of the present invention allows the reuse of block parameter data and hence reduces the memory space allocated for the block parameters in the execution of the block diagram and the generated code for the block diagram.

As defined by independent claim 16, Appellant's invention is directed to a method of processing graphical block parameters in a graphical block diagram modeling environment. In the method of the Appellant's invention, a user-defined block parameter is received, see, e.g., Specification, page 9, lines 14-18 and reference character 72 in Fig. 4, and processed to optimally produce a run-time block parameter, see, e.g., Specification, page 12, line 31 through page 13, line 12 and reference character 156 in Fig. 11. With these features, the Appellant's invention enables a graphical block diagram tool to produce optimal implementations of the block equations for a given user supplied data set. See, e.g., Specification, page 3, lines 22-28.

As defined by dependent claim 17, Appellant's invention is directed to a method of processing graphical block parameters in a graphical block diagram modeling environment, wherein a block method inversely maps the block run-time parameter to the user-defined block parameter to optimize block implementation. See, e.g., Specification, page 11, lines 1-10 and reference character 102 in Fig. 7 and Specification, page 12, line 31 through page 13, line 12 and reference character 156b in Fig. 11.

As defined by dependent claim 22, Appellant's invention is directed to a method of processing graphical block parameters in a graphical block diagram modeling environment by discarding at least a portion of the plurality of user-defined block parameters to reduce memory requirements. See, e.g., Specification, page 13, line 17-31 and reference character 168d in Fig. 12. As defined by dependent claim 23, Appellant's invention is directed to a method of mapping graphical block parameters in a graphical block diagram modeling environment, wherein like non-interfaced run-time block parameters are pooled to reduce repetition of the non-interfaced run-time block parameters. See, e.g.,

Specification, page 13, line 17-31 and reference character 168 in Fig. 12. With the pooling feature, the Appellant's invention achieves an optimal use of memory space allocated to block parameters (parameter data reuse) in both a simulation environment, as well as in automatically generated code. See, e.g., Specification, page 3, line 1-3.

As defined by independent claim 33, Appellant's invention is directed to a method of processing graphical block parameters in a graphical block diagram modeling environment. In the method of the claimed invention, user-defined block parameters are received, see, e.g., Specification, page 9, lines 14-18 and reference character 72 in Fig. 4, and processed to optimally produce run-time block parameters, see, e.g., Specification, page 12, line 31 through page 13, line 12 and reference character 156 in Fig. 11. Like non-interfaced run-time block parameters are pooled together to create a run-time parameter expression for use during modeling. See, e.g., Specification, page 13, line 17-31 and reference character 168 in Fig. 12.

As defined by independent claim 45, Appellant's invention is directed to a medium for use in a graphical modeling environment on an electronic device. The medium of the Appellant's invention holds instructions executable using an electronic device for performing a method of processing graphical block diagram block parameters. In the method, a user-defined block parameter is received, see, e.g., Specification, page 9, lines 14-18 and reference character 72 in Fig. 4, and processed to optimally produce a run-time block parameter for use during modeling, see, e.g., Specification, page 12, line 31 through page 13, line 12 and reference character 156 in Fig. 11.

As defined by independent claim 46, Appellant's invention is directed to a medium for use in a graphical modeling environment on an electronic device. The medium of the Appellant's invention holds instructions executable using an electronic device for performing the method of processing graphical block diagram block parameters. In the method, a plurality of user-defined block parameters is received, see, e.g., Specification, page 9, lines 14-18 and reference character 72 in Fig. 4, and processed to optimally produce a plurality of run-time block parameters, see, e.g., Specification, page 12, line 31 through page 13, line 12 and reference character 156 in Fig. 11. The method pools together like non-interfaced run-time block parameters to create a run-time parameter expression for use during modeling. See, e.g., Specification, page 13, line 17-31 and reference character 168 in Fig. 12.

VI. GROUNDS OF OBJECTION TO BE REVIEWED ON APPEAL

(1) Claims 16 and 45 are rejected under U.S.C. §102(e) as being anticipated by United States Patent No. 5,481,741 to McKaskle *et al.* ("McKaskle").

(2) Claims 17-20, 22-26, 33-38 and 46 are rejected under U.S.C. §103(a) as being unpatentable over McKaskle in view of United States Patent No. 6,754,885 to Dardinski *et al.* ("Dardinski").

(3) Claims 21, 27-29, 30-32 and 39-44 are rejected under U.S.C. §103(a) as being unpatentable over McKaskle in view of Dardinski and further in view of United States Patent No. 5,966,532 to McDonald *et al.*

VII. ARGUMENT

Appellant believes that the following arguments address each of the grounds of rejection to be reviewed on appeal.

1. Claims 16 and 45 are not anticipated by McKaskle because McKaskle does not disclose receiving and processing <u>a user-defined block parameter to optimally produce a run-time block parameter.</u>

Claims 16 and 45 are rejected as anticipated by United States Patent No. 5,481,741 to McKaskle *et al.* ("McKaskle"). To establish a *prima facie* case of anticipation, each and every element and limitation of the present invention must be disclosed expressly or inherently in a single prior art reference. *RCA Corp. v. Applied Digital Data Sys., Inc.*, 730 F.2d 1440, 1444, 221 USPQ 385, 388 (Fed. Cir. 1984). Appellant respectfully submits that McKaskle fails to disclose each and every element and limitation of claims 16 and 45.

Independent claims 16 and 45 are directed to a method and a medium, respectively, for processing graphical block parameters in a graphical block diagram environment. The independent claims recite, at least in part, that a user-defined block parameter is received and processed to optimally produce a run-time block parameter. A user may specify a block parameter for a block in the block diagram, such as the gain parameter in the gain block (60) depicted in Fig. 3, through a dialog box (70) shown in Fig. 4 or programmatically via a command. The user-defined block parameter may be processed by invoking a block method, such as the SetupRunTimeParams() method described in Fig. 11, to optimally produce a run-time block parameter. The run-time block parameter may include its numerical value, a mapping back to the user-defined block parameter it was created from, and its characteristic, e.g., data type, numeric type (real v. complex), and dimensions. See, e.g., Specification, page 13, lines 7-12. The run-time block parameter may be produced by, for example, pooling together like run-time parameters and/or setting up a run-time parameter execution structure that allows some variables in run-time parameter expressions to be accessed during model execution. See, e.g., Specification, page 11, lines 11-19.

In comparison, McKaskle discloses a virtual instrument including a data flow diagram and a front panel. In McKaskle, the front panel provides visual representations to the user of the input /output values to/from the data flow diagram. A user can access the front panel during the execution of the data flow diagram and change input values to the data flow diagram from the front panel. (Abstract, ll. 1-5).

McKaskle discloses in Fig. 3 a virtual instrument (40) that includes a front panel (42) and a block diagram (46) created using a front panel editor (36) and a block diagram editor (30), respectively. McKaskle discloses that the block diagram (46) receives input values from the front panel (42) and provides output values to the front panel (42). See, e.g., McKaskle, Figs. 42-43, 45-46 and 47. For example, Figs. 45 and 46 show a front panel and a block diagram, respectively, in which the user specifies the number of measurements and the delay between each measurement on the front panel depicted in Fig. 45. The block diagram illustrated in FIG. 46 collects the measurement based on the input data specified on the front panel, and passes the measurement data to an icon that graphs it on the front panel. See, e.g., McKaskle, Column 39, lines 1-19. McKaskle also discloses changing the attributes of the controls or indicators on the front panel (42) interactively or programmatically during the execution of the block diagram (46). The attributes affect the *appearance* of the controls or indicators.

The Examiner asserts in the final rejection mailed on February 17, 2005 that McKaskle discloses the receiving step of the present invention at Column 6, lines 48-49 and Column 11, lines 55-67. Final Rejection at p. 2, next to last line to p. 3, line 5. Appellant respectfully disagrees with the Examiner's assertion. In the cited excerpts McKaskle discloses a high-level diagram of the system (Fig. 2) and that "the block diagram editor 30 is used to construct and display a graphical diagram, referred to as a block diagram, which visually displays a procedure by

which a value for an input variable produces a value for one or more output variables . . . ." That is, in McKaskle, a user can construct a block diagram using a block diagram editor (30) depicted in Fig. 2, and the constructed block diagram receives an input value from the front panel and produces an output value to the front panel.

McKaskle, however, does not disclose receiving a user-defined block parameter of blocks in the block diagram. Block parameters are distinct from block inputs: Block parameters are specified when a user instantiates a block as a new instance of a block class – that is, when the user adds or modifies a block to a diagram in the block diagram environment. Inputs to a block represent data which the block may process to produce outputs, while block parameters affect how the processing is done. The output of the block may be affected both by the inputs and by block parameters. Instances associated with block classes have parameter specification interfaces, such as the dialog box depicted in Fig. 3, that allow users to define the block parameters. See, e.g., the Specification, page 8, lines 12-23. For examples, the gain parameters are the block parameters for the gain blocks (56, 60) shown in FIG. 3.

Receiving a value for an input of the block diagram from the front panel does not correspond to receiving a user-defined block parameter in the block diagram. To put it succinctly, the input controls of McKaskle set input values that are forwarded to the block diagram and do not set block parameters of the blocks in the block diagram as required by the claim.

McKaskle discloses changing the attributes of the control on the front panel interactively or programmatically during the execution of the block diagram. Although McKaskle discloses that a user can change the attributes of the controls or indicators on the front panel, McKaskle does not disclose a user defining block

parameters in the block diagram. The attributes of the controls or indicators on the front panel do not correspond to the parameters of blocks in the block diagram, because the attributes affect only the appearance (visual output) of the controls or indicators, not their behavior and/or data output.

The Examiner also asserts in the final rejection that McKaskle discloses the processing step of the present invention at Column 5, lines 47-51 and Column 6, lines 24-27. Appellant respectfully disagrees with the Examiner's assertion. One of the sections of McKaskle upon which the Examiner relies reads:

> "The present invention comprises a system and method which allows a user to programmatically access various parameters of a control or indicator. In this manner, a user can programmatically make changes that affect the output or appearance of controls and indicators."

McKaskle, Col. 5, lines 47-51, cited in Final Rejection of February 17, 2005 at page 3, lines 6-10. The other section of McKaskle upon which the Examiner relies reads:

> "The user can also view changes to the attribute during execution. Some attributes can be changed by a user and practically all can be changed by the execution subsystem."

McKaskle, Col. 6, lines 24-27, cited in Final Rejection of February 17, 2005 at page 3, lines 10-13. McKaskle addresses visualization of front panel controls or indicators, by disclosing how to use attributes of those controls in a data flow diagram or programmatically change attributes of indicators by the data flow diagram. See, e.g., McKaskle, Col 5, lines 62-67 ("more meaningful visual output to the user"; "attribute node to affect the visual output of a control provided on the front panel . . ."). The present invention, however, is directed to how user-defined parameters associated with blocks in a block diagram are processed for use in run-time. These claims are not directed to front panel controls or indicators.

In the Advisory Action mailed on June 10, 2005, the Examiner again asserts that McKaskle discloses the processing step of the present invention, citing McKaskle, Column 13, lines 18-35. In this section, McKaskle discloses that Fig. 22 shows a complete block diagram (46) of the instrumentation system (402) depicted in Fig. 5A. McKaskle also discloses that "[o]nce the block diagram 46 and front panel 42 have been created using the block diagram editor 30 and front panel editor 36, the user can then execute the block diagram (FIG. 22) using the execution subsystem 32." See, e.g., McKaskle, Column 13, lines 31-35. The Examiner asserts in the Advisory Action that the block diagram is executed to produce a run-time block parameter. According to the Examiner's assertion, a run-time block parameter is produced *by* the execution of a block diagram. In the present invention, however, the run-time block parameter is generated *before the execution* so that the run-time block parameter can be used during the execution of the block diagram. In this respect, the Examiner misapplies McKaskle to the present invention and hence the Examiner's assertion is inappropriate and does not anticipate the present invention.

Additionally, the present invention processes the user-defined block parameter to *optimally* produce a run-time block parameter. In the claimed invention, the optimal implementation of a run-time parameter can be achieved by mapping a user-defined block parameter to a run-time parameter. The mapping of the user-specified parameter to the run-time parameter enables a graphical block diagram tool to produce optimal implementations of the block equations for a given user supplied data set. See, e.g., Specification, page 3, lines 22-28.

In comparison, McKaskle discloses that a user can programmatically or interactively change the various attributes of the control on the front panel (42). McKaskle also discloses that the user can make changes that affect the output of or

appearance of the indicator on the front panel (42) during the execution of the block diagram (46). See, e.g., McKaskle, Column 5, lines 47-51. However, even if McKaskle were to disclose producing a run-time parameter, which it does not, McKaskle does not disclose processing the user-defined block parameters to *optimally* produce a run-time block parameter.

The Examiner asserts in the Advisory Action that "McKaskle allows for the user to manipulate parameters based on choices and results that the user may desire, hence allowing for processing of parameters to optimally produce a run-time block parameter." According to the Examiner assertion, the optimization process is performed in McKaskle by the *user* manipulating the parameter based on choices. However, Appellant respectfully suggests that McKaskle does not disclose or otherwise teach *optimizing* the choice nor that the choice involves producing a run-time parameter nor that it would affect run-time processing efficiency, all as addressed by the present invention. Rather, the Examiner cites material from McKaskle that addresses a "more meaningful visual output," as would be expected give McKaskle's focus on visual instrumentation and front panels. It bears quoting the section of McKaskle upon which the Examiner relies:

> "An attribute node is associated with a *control on a front panel* and operates to provide a more *meaningful visual output to the user*. The purpose of an attribute node is to affect the *visual output of a control* provided on a front panel depending on the events which occur during execution of a VI [virtual instrument] or on a user input during execution of a VI."

McKaskle, Col. 5, lines 60-66, cited in Advisory Action (emphasis added); see also Final Rejection of February 17, 2005 at p. 3 (citing McKaskle col. 5, lines 47-51). Thus, the Examiner appears to be reading "more meaningful visual output" as anticipating "optimally produce a run-time block parameter" in Claim 16. Applicant respectfully disagrees with this reading. The present invention is not

directed to visualization of input controls or output controls, but rather the method of processing a block diagram by mapping graphical block diagram block parameters to optimally produce a run-time block parameter. Similarly, that McKaskle discloses the ability to use the front panel to affect the block diagram output by using the front panel to change an input value is unremarkable and is unrelated to the present invention, which is addressed instead to, among other things, mapping block parameters of the blocks *in* the block diagram for optimal run-time processing.

In further contrast to McKaskle, the optimization of the present invention is performed programmatically by the electronic device where the present invention is performed as required by the claim language. The Examiner's assertion is inappropriate in this respect as well.

For the reasons set forth above, McKaskle fails to disclose each and every element of claims 16 and 45. Applicant therefore respectfully submits that the Examiner fails to establish a case of anticipation, that the rejection of claims 16 and 45 under 35 U.S.C. §102 should be reversed.

2. Claims 17-20, 22-26, 33-38 and 46 are not obvious over McKaskle and Dardinski because there is no motivation to combine those references and, even when combined, they fail to teach or suggest all the limitations recited by the claims.

Claims 17-20, 22-26, 33-38 and 46 are rejected under 35 U.S.C. §103(a) as allegedly being unpatentable over McKaskle in view of U.S. Patent No. 6,754,881 ("Dardinski"). To establish a *prima facie* case of obviousness, three basic criteria must be met. First, there must be some suggestion or motivation, either in the references *themselves* or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. Second,

there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations. The teaching or suggestion to make the claimed combination and the reasonable expectation of success must both be found in the prior art, not in Appellant's disclosure. *In re Vaeck*, 947 F.2d 488, 20 USPQ2d 1438 (Fed. Cir. 1991); MPEP §2142.

2.1. There is no motivation to combine <u>the teachings of McKaskle and Dardinski.</u>

Appellant submits that there is no motivation to combine the teachings of the cited prior art references.

The initial burden is on the Examiner to provide some suggestion of the desirability of doing what the inventor has done. "To support the conclusion that the claimed invention is directed to obvious subject matter, either the references must expressly or impliedly suggest the claimed invention or the examiner must present a convincing line of reasoning as to why the artisan would have found the claimed invention to have been obvious in light of the teachings of the references." *Ex parte Clapp*, 227 USPQ 972, 973 (BPAI 1985); MPEP §2142. The Examiner, however, simply asserts in the Office Action that the combination of McKaskle and Dardinski would enhance the prior art system. 'Obvious to try' does not constitute obviousness. MPEP 2145. The motivation to combine the teachings of the cited prior art references should be based on objective evidence of record. *In re Lee*, 277 F.3d 1338, 1342-44, 61 USPQ2d 1430, 1433-34 (Fed. Cir. 2002). The Examiner's assertion is subjective, but not objective. "[T]he factual question of motivation is material to patentability, and could not be resolved on subjective belief and unknown authority." *Id.*

McKaskle relates to a data flow diagram and teaches changing the attributes of the controls and indicators on the front panel. Dardinski, on the other hand, relates to a process control configuration system and teaches controlling object appearance in the data process control configuration system. There would be no need for a control panel of McKaskle in the system of Dardinski, because the process control system of Dardinski is not equipped to take additional input, such as coming from the control panel of McKaskle. Furthermore, with respect to the appearance of the system elements, Dardinski sets out a comprehensive framework for defining the appearance of configurable system components in graphical editors or other views in which the components may be depicted. See e.g. Dardinski, Abstract, lines 3-7. An artisan skilled in the art would have no motivation to combine an already flexible framework of Dardinski with the appearance-affecting attributes of McKaskle.

In light of the foregoing reasons, there is no motivation to combine the teachings of McKaskle and Dardinski. Appellant therefore submits that the Examiner fails to establish a case of obviousness, and that the rejection of claims 17-20, 22-26, 33-38 and 46 under 35 U.S.C. §103 should be reversed.

2.2. McKaskle and Dardinski fail to <u>teach or suggest all of the claim limitations.</u>

Claims 17-20, 22-26, 33-38 and 46 are rejected as obvious over McKaskle in view of Dardinski. Appellant argues above why there is no motivation to combine McKaskle and Dardinski. Even if combined, *arguendo,* however, McKaskle and Dardinski fail to teach or suggest all the claim limitations of the claimed invention.

2.2.1. Claims 18-20 and 24-26 are not obvious because combination of

McKaskle and Dardinski does not teach or suggest receiving and processing a user-defined block parameter to optimally produce a run-time block parameter.

Appellant respectfully submits that claims 18-20 and 24-26 are not obvious over McKaskle and Dardinski because those references, either alone or in combination, fail to teach or suggest all the claim limitations of the claimed invention and, therefore, fail to make a *prima facie* case of obviousness.

Claims 18-20 and 24-26 depend upon independent claim 16 and incorporate the patentable features of claim 16. Appellant submits that the combination of McKaskle and Dardinski does not teach receiving and processing the user-defined block parameter to optimally produce a run-time block parameter, as recited in claim 16.

Appellant has discussed the subject matter recited in claim 16 and the teachings of McKaskle in detail above in subsection 1. Thus, in the interest of space and brevity, Appellant will cross reference that discussion as appropriate, rather than repeat it. In light of the discussion in subsection 1, Appellant submits that the above discussion demonstrates that McKaskle does not teach receiving a user-defined block parameter of the block in the block diagram, as recited in claim 16, as the Examiner asserts in the final rejection mailed on February 17, 2005 (see Final Rejection at page 3, lines 1-5).

Furthermore, for the reasons set forth in the above discussion about claim 16 and McKaskle, see *supra* at pages 7-13, Appellant submits that McKaskle does not teach processing the user-defined block parameter to optimally produce a run-time block parameter, as recited in claim 16. Dardinski does not supplement the failing of McKaskle to teach those limitations.

Dardinski is cited by the Examiner to provide teachings for the limitations added in dependent claims. Dardinski teaches appearance objects (or other data and/or programming constructs) defining the appearance of configurable system components in graphical editors or other views in which the components may be depicted. Dardinski also teaches the configurable objects can be used to define blocks, loops and other components of a process control system. Dardinski, however, does not teach receiving and processing the user-defined block parameter to optimally produce a run-time block parameter, as recited in independent claim 16.

The combination of McKaskle and Dardinski therefore fails to teach or suggest the invention recited by independent claim 16. Claims 18-20 and 24-26 depend on independent claim 16 and are not obvious in view of the combination of McKaskle and Dardinski for at least the same reasons as above. Accordingly, Appellant respectfully submits that the Examiner fails to make a case of obviousness, and the rejection of claims 18-20 and 24-26 under 35 U.S.C. §103 should be reversed.

2.2.2. Claim 17 is not obvious because the combination of McKaskle and Dardinski does not teach mapping the run-time block parameter to the user-defined block parameter to optimize block implementation.

Appellant respectfully submits that claim 17 is not obvious over McKaskle and Dardinski because those references, even if combined, fail to teach or suggest all the claim limitations of the claimed invention and, therefore, fail to make a case of obviousness.

Claim 17 depends on claim 16 and incorporates the patentable features of claim 16. The arguments presented above with respect to the rejection of claims 18-20 and 24-26, as discussed above in subsection 2.2.1, apply with equal force here and are reiterated as if set forth in full.

Additionally, Appellant respectfully submits that McKaskle and Dardinski do not teach a block method inversely mapping the block-run-time parameters to the user-defined block parameter to optimize block implementation, as recited in claim 17. In the Office Action made Final, the Examiner cites Dardinski at Column 17, lines 42-52 and Fig. 13, as teaching this limitation. Appellant respectfully disagrees. Dardinski teaches the relationship between a Typed Object and an instance of an Object Type, and the relationship between the instance of the Object Type and a Parameterized Object which defines the Typed Object. That is, Dardinski teaches that a Typed Object references an Object Type, which references a Parameterized Object. Dardinski provides an example that if a user drags a symbolic representation of a definition (a block) and drops it into a view, this relationship provides a quick access to the Parameterized Object which actually defines the block.

In contrast, the claimed invention inversely maps the block-run-time parameters to the user-defined block parameter to optimize block implementation. In an illustrative embodiment described at page 10, line 26 through page 11, line 10 of the pending application, a parameter processor (94) makes a call to the blocks to set up run-time parameters. In response, the blocks define run-time parameters and map the run-time parameters to the user-defined parameters. Dardinski does not teach the mapping of the run-time parameters to the user-defined parameters. Rather, Dardinski just teaches that a symbolic representation

of a definition (a block) references the Parameterized Object which actually defines the block.

The combination of McKaskle and Dardinski fails to teach or suggest the invention recited by claim 17. Accordingly, Appellant respectfully submits that the Examiner fails to make a case of obviousness and the rejection of claim 17 under 35 U.S.C. §103 should be reversed.

2.2.3. Claim 22 is not obvious because combination of McKaskle and Dardinski does not teach mapping by discarding at least a portion of the plurality of user-defined block parameters to reduce memory requirements.

Appellant respectfully submits that claim 22 is not obvious over McKaskle and Dardinski because those references, either alone or in combination, fail to teach or suggest all the claim limitations of the claimed invention and, therefore, fail to make a *prima facie* case of obviousness.

Claim 22 depends on claim 16 and incorporates the patentable features of claim 16. The arguments presented above with respect to the rejection of claims 18-20 and 24-26, as discussed above in subsection 2.2.1, apply with equal force here and is reiterated as if set forth in full.

Additionally, Appellant respectfully submits that McKaskle and Dardinski do not teach mapping by discarding at least a portion of the plurality of user-defined block parameters to reduce memory requirements, as recited in claim 22. In the Office Action made Final, the Examiner cites Dardinski at Column 82, line 23 as teaching this limitation. Dardinski teaches general graphical control algorithm diagram editor functions, including graphical functions and database functions. Dardinski also teaches that the database functions can add and delete blocks to sheet. Dardinski, however, does not teach discarding at least a portion of

the plurality of user-defined block parameters to reduce memory requirements. In contrast, the present invention discards a portion of the plurality of user-defined block parameters, not the blocks. Dardinski only teaches deleting blocks.

The combination of McKaskle and Dardinski therefore fails to teach or suggest the invention recited by claim 22. Accordingly, Appellant respectfully submits that the Examiner fails to make a case of obviousness and the rejection of claim 22 under 35 U.S.C. §103 should be reversed.

2.2.4. Claim 23 is not obvious because combination of McKaskle and Dardinski does not teach pooling like non-interfaced run-time block parameters to reduce repetition of the non-interfaced run-time block parameters.

Appellant respectfully submits that claim 23 is not obvious over McKaskle and Dardinski because those references, either alone or in combination, fail to teach or suggest all the claim limitations of the claimed invention and, therefore, fail to make a *prima facie* case of obviousness.

Claim 23 depends on claim 16 and incorporates the patentable features of claim 16. The arguments presented above with respect to the rejection of claims 18-20 and 24-26, as discussed above in subsection 2.2.1, apply with equal force here and is reiterated as if set forth in full.

Additionally, Appellant submits that McKaskle and Dardinski do not teach pooling together like non-interfaced run-time block parameters to reduce repetition of the non-interfaced run-time block parameters, as recited in claim 23. The Examiner cites Dardinski at Column 11, lines 1-5 as teaching this limitation. Dardinski teaches that users can organize parameters into groups, that each group contains logically-related parameters, and that the groups can be pre-defined and/or defined by the user. In contrast, the claimed invention programmatically

pools together like non-interfaced run-time block parameters to reduce repetition of the non-interfaced run-time block parameters. Dardinski does not teach pooling together like non-interfaced run-time block parameters to reduce repetition of the non-interfaced run-time block parameters.

The combination of McKaskle and Dardinski therefore fails to teach or suggest the invention recited by claim 23. Accordingly, Appellant respectfully submits that the Examiner fails to make a case of obviousness, and that the rejection of claim 23 under 35 U.S.C. §103 should be reversed.

2.2.5. Claims 33-38 and 46 are not obvious because combination of McKaskle and Dardinski does not teach pooling together like non-interfaced run-time block parameters to reduce repetition of the non-interfaced run-time block parameters.

Appellant respectfully submits that claims 33-38 and 46 are not obvious over McKaskle and Dardinski because those references, alone or in combination, fail to teach or suggest all the claim limitations of the claimed invention and, therefore, fail to make a *prima facie* case of obviousness.

Independent claims 33 and 46 are directed to a method and a medium, respectively, of mapping graphical block diagram block parameters in a graphical block diagram modeling environment. The independent claims recite, at least in part, that user-defined block parameters are received and processed to optimally produce run-time block parameters. Like non-interfaced run-time block parameters are pooled together to create a run-time parameter expression for use during modeling. Claims 34-38 depends upon claim 33.

In the present invention, a user may specify a block parameter of blocks in the block diagram, such as the gain parameter in the gain block (60) depicted in

Fig. 3, through a dialog box (70) shown in Fig. 4 or programmatically via a command. The user-defined block parameter may be processed by invoking a method, such as the SetupRunTimeParams() method described in Fig. 11, to optimally produce a run-time block parameter. The run-time block parameter may include its numerical value, a mapping back to the user-defined block parameter it was created from, and its characteristic, e.g., data type, numeric type (real v. complex), and dimensions. See, e.g., Specification, page 13, lines 7-12. If the run-time parameter does not exist in the aggregated run-time parameter table, the run-time parameter is added to an aggregated run-time parameter table. See, e.g., Fig. 12, reference character 168c. If the run-time parameter exists in the aggregated run-time parameter table, the run-time parameter is freed. See, e.g., Fig. 12, reference character 168d.

Appellant has discussed the subject matter recited in claim 16 and the teachings of McKaskle in detail above in subsection 1. The limitations of receiving user-defined block parameters and processing the user-defined block parameters to optimally produce run-time block parameters, which are recited in claim 16, are also recited in claims 33 and 46. Thus, in the interest of space and brevity, Appellant will cross reference that discussion as appropriate, rather than repeat it. In light of the discussion in subsection 1, Appellant submits that the above discussion demonstrates that McKaskle does not teach receiving user-defined block parameters of the blocks in the block diagram, as recited in claims 33 and 46, as the Examiner asserts in the final rejection mailed on February 17, 2005 (see Final Rejection at page 4, lines 4-15).

Furthermore, for the reasons set forth in the above discussion about claim 16 and McKaskle, see *supra* at pages 7-12, Appellant submits that McKaskle does not teach processing the user-defined block parameters to optimally produce run-

time block parameters, as recited in claims 33 and 46, as Examiner asserts in the Final Rejection (at page 4, lines 4-15).

The Examiner asserts that Dardinski teaches pooling together like non-interfaced run-time block parameters to create a run-time parameter expression for use during modeling. Dardinski teaches that parameters are organized into groups and each group contains logically-related parameters. Dardinski also teaches that the groups can be pre-defined and/or defined by the user. Dardinski, however, does not teach pooling together like non-interfaced run-time block parameters to create a run-time parameter expression for use during modeling, as recited in claims 33 and 46.

In light of the foregoing reasons, the combination of McKaskle and Dardinski fails to teach the invention recited by 33 and 46. Claims 34-38, which depend upon claim 33, are not rendered obvious over the cited prior art. Accordingly, Appellant respectfully submits that the Examiner fails to make a case of obviousness and that the rejection of claims 33-38 and 46 under 35 U.S.C. §103 should be reversed.

3. Claims 21, 27-29, 30-32 and 39-44 are not obvious over McKaskle, Dardinski and McDonald because those references, even when combined, fail to teach or suggest all the limitations recited by the claims.

Claims 21, 27-29, 30-32 and 39-44 are rejected under 35 U.S.C. §103(a) as being unpatentable over McKaskle in view of Dardinski and further in view of U.S. Patent No. 5,966,532 ("McDonald").

3.1. Claims 21, 27-29 and 30-32 are not obvious over McKaskle, Dardinski and McDonald because those references, even when combined, fail to teach or suggest all the limitations recited by claim 16.

Appellant respectfully submits that claims 21, 27-29 and 30-32 are not obvious over McKaskle, Dardinski and McDonald because those references, alone or in combination, fail to teach or suggest all the claim limitations of the claimed invention and, therefore, fail to make a *prima facie* case of obviousness.

Claims 21, 27-29 and 30-32 depend upon claim 16 and incorporate the patentable features of claim 16. The arguments presented above with respect to the rejection of claims 18-20 and 24-26, as discussed above in subsections 2.1 and 2.2.1, apply with equal force here and are reiterated as if set forth in full.

Appellant submits that McDonald also fails to teach receiving and processing the user-defined block parameter to optimally produce a run-time block parameter, as recited in claim 16. McDonald is cited by the Examiner to provide teachings for the limitations added in dependent claims. McDonald relates to generating graphical code in a graphical programming system, more specifically a data flow diagram is generated from a wizard associated with a separate front panel. McDonald teaches the graphical programming system includes a plurality of front panel objects or controls which represent the user interface. McDonald also teaches that the user can select parameter values in the wizard to configure certain aspects of the graphical code being created. See, e.g., McDonald, Column 11, lines 42-55. McDonald further teaches that the graphical code generation wizard selects a graphical code template in response to the control and configures the graphical code template with the parameter values. McDonald, however, does not teach processing the user-defined block parameter to optimally produce a run-time block parameter, as recited in claim 16.

The combination of McKaskle, Dardinski and McDonald therefore fails to teach or suggest all of the limitations of claim 16. Claims 21, 27-29 and 30-32,

which depend upon claim 16, are not rendered obvious over the cited prior art. Accordingly, Appellant respectfully submits that the Examiner fails to make a case of obviousness, and that the rejection of claims 21, 27-29 and 30-32 under 35 U.S.C. §103 should be reversed.

3.2. Claims 39-44 are not obvious over McKaskle, Dardinski and McDonald because those references, even when combined, fail to teach or suggest all the limitations recited by claim 33.

Appellant respectfully submits that claims 39-44 are not obvious over McKaskle, Dardinski and McDonald because those references, alone or in combination, fail to teach or suggest all the claim limitations of the claimed invention and, therefore, fail to make a *prima facie* case of obviousness.

Claims 39-44 depend upon claim 33 and incorporate the patentable features of claim 33. The arguments presented above with respect to the rejection of claim 33, as discussed above in subsections 2.1 and 2.2.5, apply with equal force here and are reiterated as if set forth in full.

Appellant submits that McDonald also fails to teach receiving and processing the user-defined block parameter to optimally produce a run-time block parameter, and pooling together like non-interfaced run-time block parameters to create a run-time parameter expression for use during modeling, as recited in claim 33. McDonald is cited by the Examiner to provide teachings for the limitations added in dependent claims. McDonald relates to generating graphical code in a graphical programming system, more specifically a data flow diagram is generated from a wizard associated with a separate front panel. McDonald teaches the graphical programming system includes a plurality of front panel objects or controls which represent the user interface. McDonald also teaches that the user can select parameter values in the wizard to configure certain aspects of the

graphical code being created. See, e.g., McDonald, Column 11, lines 42-55. McDonald further teaches that the graphical code generation wizard selects a graphical code template in response to the control and configures the graphical code template with the parameter values. McDonald, however, does not teach receiving processing the user-defined block parameter to optimally produce a run-time block parameter, as recited in claim 33. McDonald also does not teach pooling together like non-interfaced run-time block parameters to create a run-time parameter expression for use during modeling, as recited in claim 33.

The combination of McKaskle, Dardinski and McDonald therefore fails to teach all of the limitations of claim 33. Claims 39-44, which depend upon claim 33, are not rendered obvious over the cited prior art. Accordingly, Appellant respectfully submits that the Examiner fails to make a case of obviousness and that the rejection of claims 39-44 under 35 U.S.C. §103 should be reversed.

XIII. CLAIMS APPENDIX

A copy of the claims involved in the present appeal is attached hereto as Appendix A. As indicated above, the claims in Appendix A include the amendments filed by Appellant on July 16, 2004.

IX. EVIDENCE APPENDIX

No evidence pursuant to §§ 1.130, 1.131, or 1.132 or entered by or relied upon by the examiner is being submitted.

X. RELATED PROCEEDINGS APPENDIX

No related proceedings are referenced in II. above and there are no copies of decisions to be provided

Dated: **November 15, 2005**

Respectfully submitted,

By ____________________

Kevin J. Canning
Registration No.: 35,470
LAHIVE & COCKFIELD, LLP
28 State Street
Boston, Massachusetts 02109
(617) 227-7400
(617) 742-4214 (Fax)
Attorney/Agent For Appellant

APPENDIX A

Claims Involved in the Appeal of Application Serial No. 09/911663

16. A method of mapping graphical block diagram block parameters in a graphical block diagram modeling environment, comprising:

receiving a user-defined block parameter; and

processing the user-defined block parameter to optimally produce a run-time block parameter.

17. The method of claim 16, further comprising a block method inversely mapping the block run-time parameter to the user-defined block parameter to optimize block implementation.

18. The method of claim 16, further comprising receiving a plurality of user-defined block parameters.

19. The method of claim 18, further comprising processing the plurality of user-defined block parameters to optimally produce a run-time block parameter.

20. The method of claim 19, wherein the plurality of user-defined block parameters is processed to produce a single run-time block parameter.

21. The method of claim 19, wherein the run-time block parameter is configured to return at least one of simulation results, and automatically generated code that implements graphical block diagram model equations.

22. The method of claim 19, further comprising mapping by discarding at least a portion of the plurality of user-defined block parameters to reduce memory requirements.

23. The method of claim 19, further comprising pooling like non-interfaced run-time block parameters to reduce repetition of the non-interfaced run-time block parameters.

24. The method of claim 23, wherein the pooling step comprises mapping user-defined block parameters into an existing pool.

25. The method of claim 23, wherein the pooling step is repeated with additional optimization.

26. The method of claim 19, further comprising mapping by translating the plurality of user-defined block parameters based at least in part on type.

27. The method of claim 16, wherein the run-time block parameter is configured to return at least one of simulation results, and automatically generated code that implements graphical block diagram model equations.

28. The method of claim 27, wherein when the code is automatically generated, the parameter expressions are maintained in the automatically generated code.

29. The method of claim 28, wherein the parameter expressions contain interfaced variables that are updatable during modeling.

30. The method of claim 29, further comprising converting to a relatively more compact representation portions of the parameter expressions that are constants while providing access to interfaced variables that are updatable.

31. The method of claim 29, wherein interfaced variables are updatable.

32. The method of claim 31, wherein updatable variables used in a plurality of blocks are declared only once.

33. A method of mapping graphical block diagram block parameters in a graphical block diagram modeling environment, comprising:

receiving a plurality of user-defined block parameters;

processing the plurality of user-defined block parameter to optimally produce a plurality of run-time block parameters;

pooling together like non-interfaced run-time block parameters to create a run-time parameter expression for use during modeling.

34. The method of claim 33, wherein pooling further comprises mapping user-defined block parameters into an existing pool.

35. The method of claim 33, wherein the non-interfaced run-time block parameters have stored values that differ from presented values.

36. The method of claim 35, wherein the non-interfaced run-time block parameters are fixed point.

37. The method of claim 33, further comprising translating at run-time constant parameter values to an internal representation to enable increased pooling.

38. The method of claim 33, wherein the step of pooling further comprises collecting constant portions of an expression containing an interfaced variable.

39. The method of claim 33, wherein the run-time block parameters are configured to return at least one of simulation results, and automatically generated code that implements graphical block diagram model equations.

40. The method of claim 39, wherein when the code is automatically generated, the parameter expressions are maintained in the automatically generated code.

41. The method of claim 40, wherein the parameter expressions contain interfaced variables which are updatable.

42. The method of claim 41, further comprising converting to a relatively more compact representation portions of the parameter expressions that are constants while providing access to interfaced variables.

43. The method of claim 41, wherein interfaced variables are updatable.

44. The method of claim 43, wherein updatable variables used in a plurality of blocks are declared only once.

45. A medium for use in a graphical modeling environment on an electronic device, the medium holding instructions executable using the electronic device for

performing a method of mapping graphical block diagram block parameters, the method comprising:

receiving a user-defined block parameter; and

processing the user-defined block parameter to optimally produce a run-time block parameter for use during modeling.

46. A medium for use in a graphical modeling environment on an electronic device, the medium holding instructions executable using the electronic device for performing a method of mapping graphical block diagram block parameters, the method comprising:

receiving a plurality of user-defined block parameters;

processing the plurality of user-defined block parameter to optimally produce a plurality of run-time block parameters; and

pooling together like non-interfaced run-time block parameters to create a run-time parameter expression for use during modeling.